

Appendix A

**Inventors: Dan Bartell
Richard Watts**

**Attorney Docket Number: 3272
Title: Methods, Systems and Software for Pixel Stutter Detection**

```
//////////  
void CDatToolDlg::TagStutter()  
{  
  
    //Create output dat file  
    ImageData * pOutputImg;  
    pOutputImg = new ImageData;  
    pOutputImg->SetName(this->m_OutputFile);  
    pOutputImg->Create(m_pInputImage->xSize(),m_pInputImage->ySize());  
    pOutputImg->SetSize(m_pInputImage->xSize(),m_pInputImage->ySize());  
  
    //Copy input dat file to output dat file  
    this->CopyImage(pOutputImg,m_pInputImage);  
  
    //A few local variables  
    unsigned short * inputDataPointer=m_pInputImage->GetDataPtr();  
    unsigned short * outputDataPointer=pOutputImg->GetDataPtr();  
    int numCols=m_pInputImage->xSize();  
    int nStutter=0,iRowStart,iRowEnd,iColStart,iColEnd;  
  
    if (m_StutterArea == ENTIRE_IMAGE)  
    {  
        iRowStart=0;  
        iRowEnd=m_pInputImage->ySize()-1;  
        iColStart=0;  
        iColEnd=m_pInputImage->xSize()-1;  
    }  
    else  
    {  
        //Does the input image have a valid grid?  
        if ( (m_pInputImage->CellGrid.lowerleft.y > 0) && (m_pInputImage->CellGrid.lowerleft.x > 0) )  
        {  
            //yes, valid grid  
  
            iRowStart=__max(m_pInputImage->CellGrid.upperleft.y,m_pInputImage->CellGrid.upperright.y);  
            iRowEnd= __min(m_pInputImage->CellGrid.lowerleft.y,m_pInputImage->CellGrid.lowerright.y);  
  
            iColStart=__max(m_pInputImage->CellGrid.upperleft.x,m_pInputImage->CellGrid.lowerleft.x);  
            iColEnd=__min(m_pInputImage->CellGrid.upperright.x,m_pInputImage->CellGrid.lowerright.x);  
        }  
        else  
        {  
            //no, invalid grid  
            AfxMessageBox("Operation Canceled due to invalid grid");  
  
            //Dispose memory  
            delete pOutputImg;  
            pOutputImg=NULL;  
  
            return;  
        }  
    }  
}
```

```

if (m_StutterDirection==HORIZONTAL_STUTTER)
{
    //For all the image pixels
    for (int iRow=iRowStart;iRow <=iRowEnd;iRow++)
    {
        //Get A row of the input image
        int iRowOffset=iRow * numCols;

        //Get the first pixel value in the row
        inputDataPointer=m_pInputImage->GetDataPtr();
        unsigned short iPixValue=inputDataPointer[iRowOffset+iColStart];

        //Walk across the row
        for ( int iCol=iColStart+1;iCol <= iColEnd;iCol++)
        {
            //If the pixel is identical to the next pixel, then set both pixels to the
            //stutter replacement value
            if (inputDataPointer[iRowOffset + iCol] == iPixValue)
            {
                nStutter++;
                inputDataPointer[iRowOffset + iCol] = m_StutterReplace;
                inputDataPointer[iRowOffset + iCol-1] = m_StutterReplace;
            }
            else
            {
                //do nothing
            }

            //move on
            iPixValue=inputDataPointer[iRowOffset+iCol];
        }
    }
}
else
{
    //Vertical stutter
    //For all the image pixels
    for (int iCol=iColStart;iCol<=iColEnd;iCol++)
    {
        //For each column, walk the row
        int iOffsetPrev=iRowStart*numCols + iCol;
        unsigned short iPixValuePrev = inputDataPointer[iOffsetPrev];

        //Walk down the column
        for (int iRow=iRowStart+1; iRow <= iRowEnd; iRow++)
        {
            int iOffset=iRow*numCols + iCol;
            unsigned short iPixValue=inputDataPointer[iOffset];
            //If the pixel is identical to the previous pixel value, then set both pixels to
            //the stutter replacement value
            if (iPixValue == iPixValuePrev)
            {
                nStutter++;
                inputDataPointer[iOffsetPrev]=m_StutterReplace;
                inputDataPointer[iOffset]=m_StutterReplace;

            }
            else
            {
                //do nothing
            }

            //move on
            iOffsetPrev=iOffset;
            iPixValuePrev=iPixValue;
        }
    }
}
}

```

```
//Write the dat file.
if (!pOutputImg->Write())
{
    AfxMessageBox(IDS_ERRORWRITE);
}
long lSize=m_pInputImage->xSize()*m_pInputImage->ySize();
long replacedPixels=2*nStutter;
float stutterRatio=(float(nStutter))/float(lSize);

//Report
CString outString1,outString2;
outString1.Format("Stutter Report for %s\nOutput Image: %s\n",m_pInputImage->GetName(),pOutputImg->GetName());

if (m_StutterArea==ENTIRE_IMAGE)
{
    outString2="Area Analyzed is Entire Image\n";
}
else
{
    outString2.Format("Area Analyzed is Rows: %d to %d and Cols: %d to %d\n",iRowStart,iRowEnd,iColStart,iColEnd);
}
outString1 += outString2;

if (m_StutterDirection == HORIZONTAL_STUTTER)
{
    outString2="Direction is Horizontal\n";
}
else
{
    outString2="Direction is Vertical\n";
}
outString1 += outString2;

outString2.Format("Total Pixels:\t%d\nReplaced Pixels:\t%d\nStutter Count:\t%d\nStutterRatio:\t%10.6f,%e",
lSize,replacedPixels,nStutter,stutterRatio,stutterRatio);

outString1 += outString2;
AfxMessageBox(outString1);

//Dispose memory
delete pOutputImg;
pOutputImg=NULL;

}
```

United States Patent & Trademark Office
Office of Initial Patent Examination

Application papers not suitable for publication

SN 09/134751 Mail Date 12-11-00

Non-English Specification

Specification contains drawing(s) on page(s) _____ or table(s) 19,

Landscape orientation of text Specification Claims Abstract

Handwritten Specification Claims Abstract

More than one column Specification Claims Abstract

Improper line spacing Specification Claims Abstract

Claims not on separate page(s)

Abstract not on separate page(s)

Improper paper size -- Must be either A4 (21 cm x 29.7 cm) or 8-1/2"x 11"
 Specification page(s) _____ Abstract
 Drawing page(s) _____ Claim(s)

Improper margins
 Specification page(s) _____ Abstract
 Drawing page(s) _____ Claim(s)

Not reproducible Section
Reason Specification page(s) _____
 Paper too thin Drawing page(s) _____
 Glossy pages Abstract
 Non-white background Claim(s)

Drawing objection(s)
 Missing lead lines, drawing(s) _____
 Line quality is too light, drawing(s) _____
 More than 1 drawing and not numbered correctly
 Non-English text, drawing(s) _____
 Excessive text, drawing(s) _____
 Photographs capable of illustration, drawing(s) _____